BUUCTF Reverse/[ACTF新生赛2020]Oruga

原创

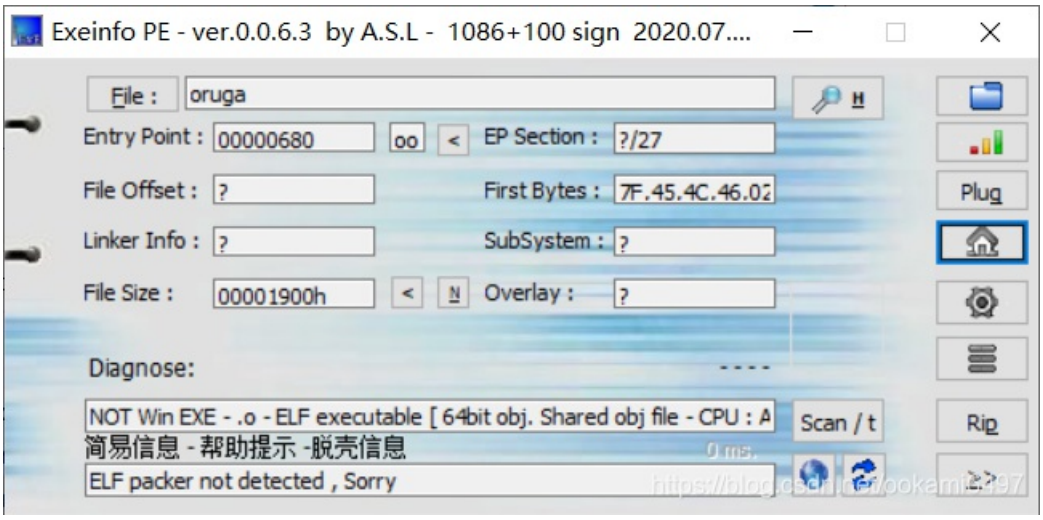
这就是强者的世界么 于 2021-07-27 15:55:57 发布 100 收藏

分类专栏： # BUUCTF Reverse

本文链接：https://blog.csdn.net/ookami6497/article/details/119145213

BUUCTF Reverse/[ACTF新生赛2020]Oruga

> 先看文件信息：没有加壳



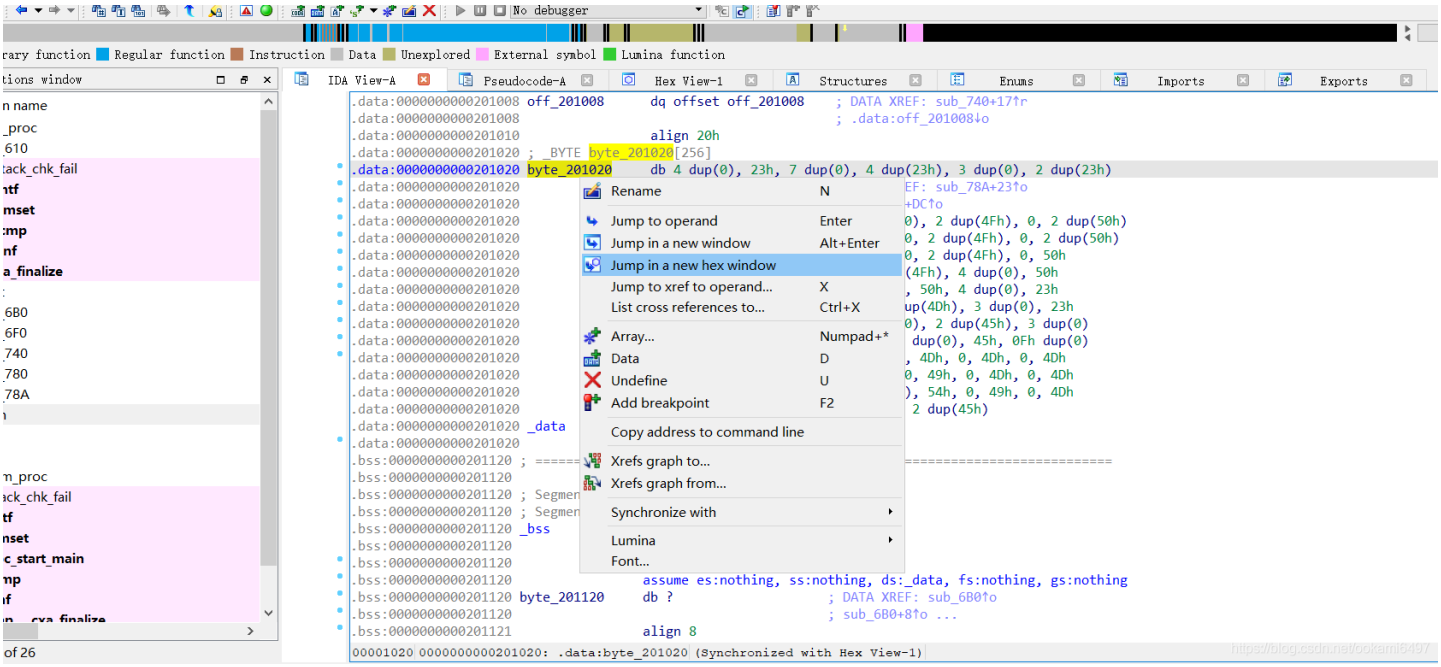
IDA64位打开，分析代码

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  __int64 result; // rax
  int i; // [rsp+0h] [rbp-40h]
  char s1[6]; // [rsp+4h] [rbp-3Ch] BYREF
  char s2[6]; // [rsp+Ah] [rbp-36h] BYREF
  char s[40]; // [rsp+10h] [rbp-30h] BYREF
  unsigned __int64 v8; // [rsp+38h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  memset(s, 0, 25uLL);
  printf("Tell me the flag:");
  scanf("%s", s);
  strcpy(s2, "actf{");
  for ( i = 0; i <= 4; ++i )
    s1[i] = s[i];
  s1[5] = 0;
  if ( !strcmp(s1, s2) )
  {
    if ( sub_78A((__int64)s) )
      printf("That's True Flag!");
    else
      printf("don't stop trying...");
    result = 0LL;
  }
  else
  {
    printf("Format false!");
    result = 0LL;
  }
  return result;
}
```

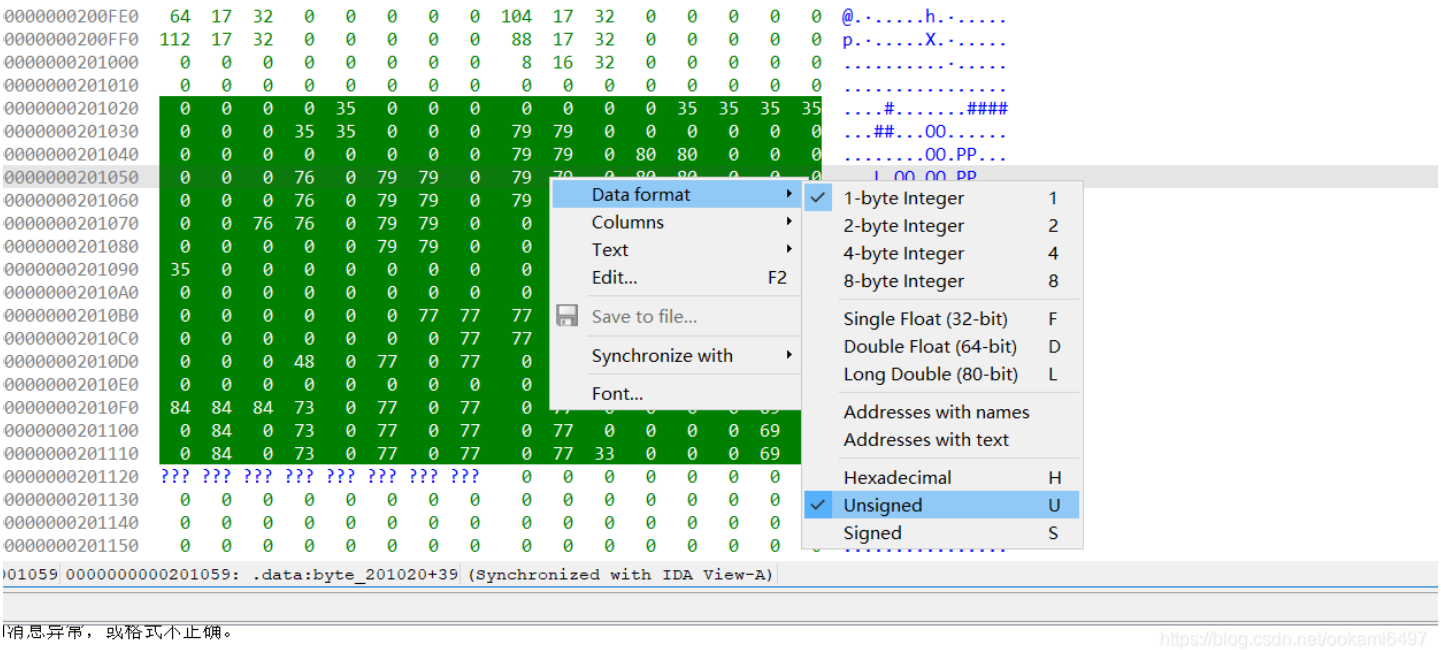
重点是if语句中的条件`if ( sub_78A((__int64)s) )`，跟进查看

```
_BOOL8 __fastcall sub_78A(__int64 a1)
{
  int v2; // [rsp+Ch] [rbp-Ch]
  int v3; // [rsp+10h] [rbp-8h]
  int v4; // [rsp+14h] [rbp-4h]

  v2 = 0;
  v3 = 5;
  v4 = 0;
  while ( byte_201020[v2] != 0x21 )
  {
    v2 -= v4;
    if ( *(_BYTE *)(v3 + a1) != 'W' || v4 == -16 )
    {
      if ( *(_BYTE *)(v3 + a1) != 'E' || v4 == 1 )
      {
        if ( *(_BYTE *)(v3 + a1) != 'M' || v4 == 16 )
        {
          if ( *(_BYTE *)(v3 + a1) != 'J' || v4 == -1 )
            return 0LL;
          v4 = -1;
        }
        else
        {
          v4 = 16;
        }
      }
      else
      {
        v4 = 1;
      }
    }
    else
    {
      v4 = -16;
    }
    ++v3;
    while ( !byte_201020[v2] )
    {
      if ( v4 == -1 && (v2 & 0xF) == 0 )
        return 0LL;
      if ( v4 == 1 && v2 % 16 == 15 )
        return 0LL;
      if ( v4 == 16 && (unsigned int)(v2 - 240) <= 0xF )
        return 0LL;
      if ( v4 == -16 && (unsigned int)(v2 + 15) <= 0x1E )
        return 0LL;
      v2 += v4;
    }
  }
  return *(_BYTE *)(v3 + a1) == '}';
}
```

> 跟进byte_201020，看到老长一串，转到hex表

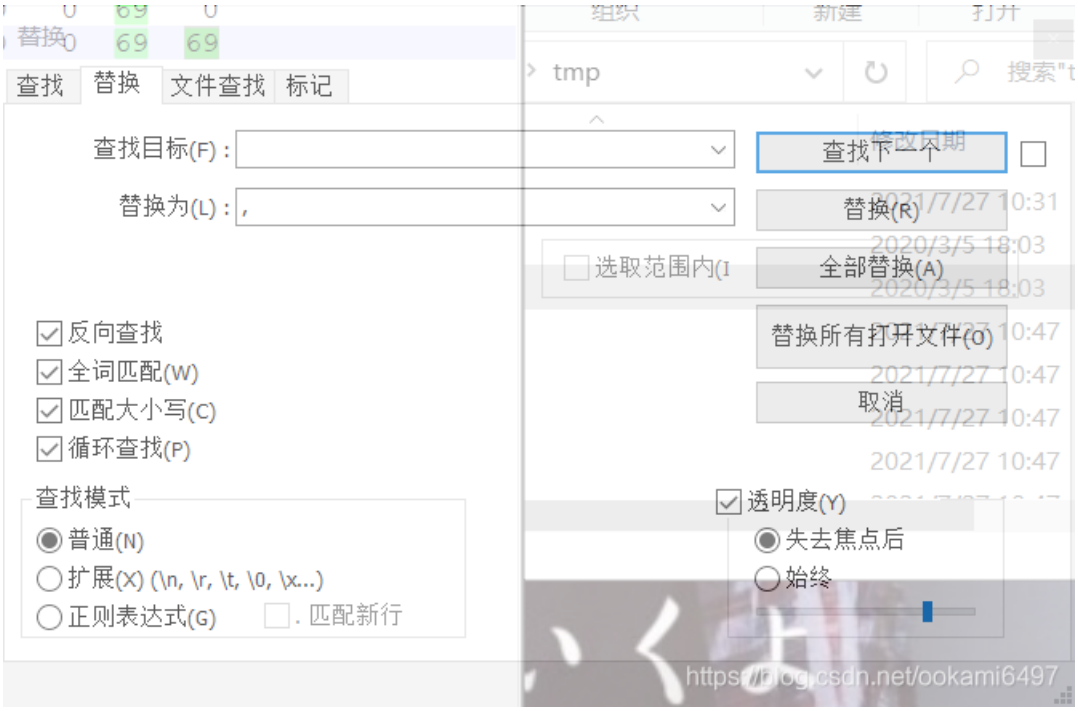


转成10进制



消息异常，或格式不正确。

把空格替换为逗号



得到

```
0,0,0,0,35,0,0,0, 0,0,0,0,35,35,35,35,
0,0,0,35,35,0,0,0,79,79,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,79,79,0,80,80,0,0,0,
0,0,0,76,0,79,79,0,79,79,0,80,80,0,0,0,
0,0,0,76,0,79,79,0,79,79,0,80,0,0,0,0,
0,0,76,76,0,79,79,0, 0,0,0,80,0,0,0,0,
0,0,0,0,0,79,79,0, 0,0,0,80,0,0,0,0,
35,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0, 0,0,0,0,35,0,0,0,
0,0,0,0,0,0,77,77,77,0,0,0,35,0,0,0,
0,0,0,0,0,0,0,77,77,77,0,0,0,0,69,69,
0,0,0,48,0,77,0,77, 0,77,0,0,0,0,69,0,
0,0,0,0,0,0,0,0, 0,0,0,0,0,0,69,69,
84,84,84,73,0,77,0,77, 0,77,0,0,0,0,69,0,
0,84,0,73,0,77,0,77, 0,77,0,0,0,0,69,0,
0,84,0,73,0,77,0,77, 0,77,33,0,0,0,69,69,
```

然后我看代码看了半天，半天没有思路，，，看了大佬的wp才知道这玩意原来是个迷宫。。。

写个脚本输出一下

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int v2 = 0;
int v4 = 0;


int main()
{
    int i,j;
    char maze[] = {0,0,0,0,35,0,0,0, 0,0,0,0,35,35,35,35,
              0,0,0,35,35,0,0,0,79,79,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,79,79,0,80,80,0,0,0,
0,0,0,76,0,79,79,0,79,79,0,80,80,0,0,0,
0,0,0,76,0,79,79,0,79,79,0,80,0,0,0,0,
0,0,76,76,0,79,79,0, 0,0,0,80,0,0,0,0,
0,0,0,0,0,79,79,0, 0,0,0,80,0,0,0,0,
35,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0, 0,0,0,0,35,0,0,0,
0,0,0,0,0,0,77,77,77,0,0,0,35,0,0,0,
0,0,0,0,0,0,0,77,77,77,0,0,0,0,69,69,
0,0,0,48,0,77,0,77, 0,77,0,0,0,0,69,0,
0,0,0,0,0,0,0,0, 0,0,0,0,0,0,69,69,
84,84,84,73,0,77,0,77, 0,77,0,0,0,0,69,0,
0,84,0,73,0,77,0,77, 0,77,0,0,0,0,69,0,
0,84,0,73,0,77,0,77, 0,77,33,0,0,0,69,69};

   for(i = 0 ; i < 16 ;i++ )
   {

       for(j = 0 ; j < 16 ; j++)
       {
           if(maze[i * 16 + j] == 0)
              printf(" ");
           else if(maze[i * 16 + j] == 33)
              printf("#");
           else printf("*");
       }
       printf("\n");
   }

   return 0;
}
```

> 得到迷宫地图，起始点在左上角！，终点是#

```
I   *       ****
   **    **
         ** **
   * ** ** **
   * ** ** *
  ** **     *
     **     *
*
             *
      ***    *
        ***     **
   * * * *     *
                **
**** * * *     *
 * * * * *     *
 * * * * *#    **
```

然后这个在空格的位置可以随便走，但是一定要碰到障碍物才能停下来，像块肥皂一样。

根据这个，可以推出
M是下
W是上
J是左
E是右

```
while ( !byte_201020[v2] )       //在空格的地方一直走，直到撞到障碍物
  {
    if ( v4 == -1 && (v2 & 0xF) == 0 )
      return 0LL;
    if ( v4 == 1 && v2 % 16 == 15 )
      return 0LL;
    if ( v4 == 16 && (unsigned int)(v2 - 240) <= 0xF )    //加减16相当于上下，最终是以v2的值决定决定是否结束
      return 0LL;
    if ( v4 == -16 && (unsigned int)(v2 + 15) <= 0x1E )
      return 0LL;
    v2 += v4;
  }
```

然后就照着迷宫写出路径得到MEWEMEWJMEWJM

flag： flag{MEWEMEWJMEWJM}